

10/532643

(12) NACH DEM VERFAHREN ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES
PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro



(43) Internationales Veröffentlichungsdatum
13. Mai 2004 (13.05.2004)

PCT

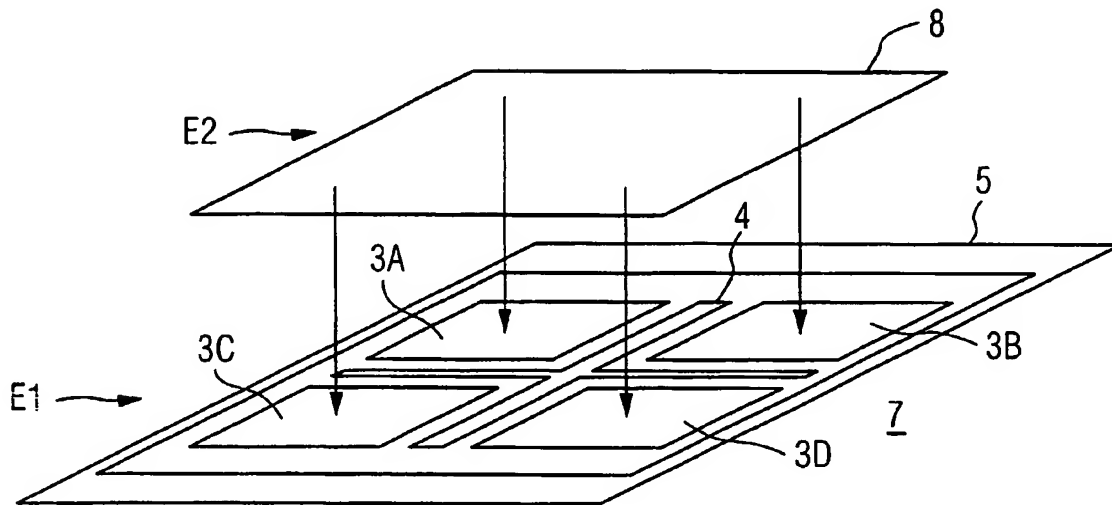
(10) Internationale Veröffentlichungsnummer
WO 2004/040766 A2

- (51) Internationale Patentklassifikation⁷: **H03K 19/177**
- (21) Internationales Aktenzeichen: **PCT/DE2003/003524**
- (22) Internationales Anmeldedatum:
23. Oktober 2003 (23.10.2003)
- (25) Einreichungssprache: **Deutsch**
- (26) Veröffentlichungssprache: **Deutsch**
- (30) Angaben zur Priorität:
102 49 676.5 24. Oktober 2002 (24.10.2002) **DE**
- (71) Anmelder (für alle Bestimmungsstaaten mit Ausnahme von
US): **SIEMENS AKTIENGESELLSCHAFT [DE/DE];**
Wittelsbacherplatz 2, 80333 München (DE).
- (72) Erfinder; und
- (75) Erfinder/Anmelder (nur für US): **SIEMERS, Christian**
[DE/DE]; Mary-Cassatt-Ring 38, 38446 Wolfsburg (DE).
- (74) Gemeinsamer Vertreter: **SIEMENS AKTIENGE-
SELLSCHAFT;** Postfach 22 16 34, 80506 München
(DE).
- (81) Bestimmungsstaaten (national): **CN, JP, KR, US.**
- (84) Bestimmungsstaaten (regional): **europäisches Patent (AT,
BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR,
HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR).**
- Veröffentlicht:**
— *ohne internationalen Recherchenbericht und erneut zu ver-
öffentlichen nach Erhalt des Berichts*

[Fortsetzung auf der nächsten Seite]

(54) Title: **PROGRAMMABLE LOGIC DEVICE**

(54) Bezeichnung: **EINRICHTUNG DER PROGRAMMIERBAREN LOGIK**



(57) **Abstract:** The invention relates to a programmable logic device (7) comprising several logic blocks (3A to 3D) with configurable characteristics, elements for linking the logic blocks to one another and a processing unit (4) and an input/output unit (5). In addition, the linking elements have at least one configurable changeover logic block, enabling the logic blocks (3A to 3D) to be re-configured during the operation of the logic device (7). Said changeover logic block is used for the configuration of at least one of the re-configurable logic blocks (3A to 3D) and its connection to other blocks and/or its connection to the processing unit (4) and/or to the input/output unit (5).

(57) **Zusammenfassung:** Die Einrichtung (7) der programmierbaren Logik umfasst mehrere Logikblöcke (3A bis 3D) mit konfigurierbaren Eigenschaften und Mittel zum Verknüpfen der Logikblöcke untereinander und mit einer Verarbeitungseinheit (4) und einer Ein-/Ausgabeeinheit (5). Eine Rekonfigurierbarkeit der Logikblöcke (3A bis 3D) während des Betriebs der Logikeinrichtung (7) ist dadurch gegeben, dass die Verknüpfungsmittel zusätzlich wenigstens einen konfigurierbaren Umschalt-Logikblock (8) aufweisen, mit dem die Konfiguration wenigstens einiger der rekonfigurierbaren Logikblöcke (3A bis 3D) untereinander und/oder mit der Verarbeitungseinheit (4) und/oder der Ein-/Ausgabeeinheit (5) erfolgt.

BEST AVAILABLE COPY

WO 2004/040766 A2



Zur Erklärung der Zweibuchstaben-Codes und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

Beschreibung

Einrichtung der programmierbaren Logik

5 Die Erfindung bezieht sich auf eine Einrichtung der programmierbaren Logik mit mehreren Logikblöcken mit konfigurierbaren Eigenschaften, die jeweils mindestens eine logische Verarbeitungseinheit mit Funktionsprogrammen und Schnittstellen zu den jeweils anderen Logikblöcken umfassen, mit mindestens
10 einer den Logikblöcken zugeordneten Eingabe- und Ausgabeeinheit sowie mit Mitteln zum Verknüpfen der Logikblöcke untereinander, mit der mindestens einen Verarbeitungseinheit eines anderen Logikblocks und mit der mindestens einen Ein-/Ausgabeeinheit. Eine derartige Logikeinrichtung ist der
15 US 4,870,302 A zu entnehmen.

Programmierbare logische Bausteine von herkömmlichen Logikeinrichtungen wie insbesondere Prozessoren führen Programme aus, die aus einem Speicher geladen werden. Die auszuführende
20 Software in Form von Befehlswörtern ist dort als Maschinenbefehl abgelegt. Diese Befehlswörter werden geladen, analysiert, interpretiert und in einer Verarbeitungseinheit ausgeführt. Dabei löst die Verarbeitung eines einzigen Befehlswortes eine Vielzahl von Einzelaktionen in der Logikeinrichtung
25 aus.

Die Basisstruktur und -organisation bekannter digitaler Logikeinrichtungen, insbesondere von Computern mit Mikroprozessoren, beruht auf der Konzeption des sogenannten "Von-
30 Neumann-Rechners". Dessen Zentraleinheit CPU ("Central Processing Unit"), d.h. dessen Computerkern, umfasst in ihrer Minimalkonfiguration als Hauptbestandteile einen Hauptspeicher, eine Steuereinheit und eine Verarbeitungseinheit (bzw. Rechenwerk):
35 - Der Hauptspeicher speichert Befehlswörter (Programmdaten) und Verarbeitungsdaten (Operantenwörter) und stellt diese auf Aufforderung zur Verfügung. Ferner nimmt der Haupt-

speicher Zwischen- und Endresultate der Verarbeitung auf. Hauptspeicher können durch flüchtige oder nicht-flüchtige Speicher realisiert werden.

- 5 - Die Steuereinheit organisiert die Reihenfolge, in der Befehls-
wörter abgearbeitet werden. Sie fordert Befehls-
wörter aus dem Hauptspeicher an und veranlasst deren Ausführung
in der Verarbeitungseinheit. Außerdem analysiert sie die
Befehls-
wörter und veranlasst die Lieferung von Verarbei-
tungsdaten an die Verarbeitungseinheit.
- 10 - Die Verarbeitungseinheit führt die Operation an den Verar-
beitungsdaten aus und liefert entsprechende Resultatwörter
an den Hauptspeicher. Für jede Operation enthält die Ver-
arbeitungseinheit ein Mikroprogramm, das die benötigten
Übertragungswege freischaltet. Die Verarbeitungseinheit
15 wird durch die Steuereinheit auf die jeweilige Operations-
art, d.h. auf den abzuarbeitenden Befehl, eingestellt.
Der Zentraleinheit sind Peripheriegeräte zugeordnet, bei
denen es sich um externe Speicher sowie um Ein- und Ausga-
begeräte handeln kann. Die angegebenen Hauptkomponenten
20 der Zentraleinheit können physikalisch getrennt sein; zu-
meist sind sie jedoch auf einem gemeinsamen Prozessorchip
mit einem Cache oder beispielsweise einem embedded ROM re-
alisiert.
- 25 Eine genauere Betrachtung der Programmierung einer solchen
programmierbaren Logikeinrichtung PLD ("Programmable Logic
Device") mit den erwähnten Grundkomponenten des Von-Neumann
Rechners zeigt, dass das Programm sowie die Initialisierungs-
daten im preiswerten Speicher (mit minimal 1 Transistor pro
30 Speicherzelle) untergebracht sind und erst dann in die CPU
zur Ausführung gelangen, wenn sie an der Reihe sind. Die
preiswerte Speicherung einerseits und die "Wiederverwendung"
der aufwendigeren CPU-Schaltkreise wie ALU ("Arithmetic Logic
Unit") für alle möglichen Instruktionen andererseits sind die
35 positiven Beiträge zu einer sogenannten "funktionalen Dichte"
der Prozessor-basierten Rechner. Hierbei ergibt die zeitliche
Sequenz der Bearbeitung natürlich einen negativen Beitrag.

Die funktionale Dichte kann dabei als mittlere Anzahl von aktiven Gatteräquivalenten pro Siliziumfläche und Zeit definiert werden.

- 5 Bei bekannten programmierbaren Logikeinrichtungen (PLDs) mit Rechenkapazität, Speicher und Ein- und Ausgabeeinheit I/O ("Input/Output"), wie sie z.B. aus der eingangs genannten US-A-Schrift zu entnehmen ist, wird deren Struktur einmal programmiert, und der Programm-(wie Daten-)Inhalt wird an der
- 10 Struktur selbst gespeichert. Diese Form der Speicherung ist aufwendig, denn um den Speicher selbst müssen zwecks schneller Umsetzung in geschaltete Datenpfade eine Menge von zusätzlichen Transistoren hinzukommen. Es existieren zwar nur wenige Angaben über die Ausnutzung des Siliziums; man sollte
- 15 von Faktoren 20 bis 40 im Verhältnis Gesamtzahl/sichtbarer Transistorkapazität (im Sinne der Schaltfunktion) ausgehen. Bei der bekannten Ausführungsform einer Logikeinrichtung bezieht sich die Verknüpfung ihrer Logikblöcke erstens auf die Kopplung von Datenausgängen auf Dateneingänge (Routing-
- 20 Verknüpfung) und zweitens auf die Verarbeitung der Eingangsdaten zu den Ausgangsdaten in den einzelnen Logikblöcken (Logik-Verknüpfung). Die Routing-Verknüpfung gilt sowohl für Daten, die aus Logikblöcken (Logic Elements) stammen bzw. in diese geführt werden, als auch für solche, die aus I/O-Pads
- 25 stammen bzw. in diese geleitet werden. In diesem System der bekannten Ausführungsform ist ausschließlich an eine Datenkopplung gedacht.

- Aus der US 6,333,641 B1 geht eine programmierbare Logikein-
- 30 richtung mit einem Array von Logikmodulen oder -blöcken hervor. Eine Verbindungseinheit mit vertikalen Routing(Leitweg-)Bahnen, horizontalen Routing-Bahnen und lokalen Routing-Bahnen verknüpft die Logikblöcke. Ein Omni-(universeller)Bus (Datenaustauschsammeleisenbahn) ist über das Array gelegt, der
- 35 mit dem Array derart verknüpft ist, dass dieser dynamisch selbständige Sub-Arrays der Logikblöcke mit variabler Größe bildet, die ihrerseits mit dem Omni-Bus verbunden sind. Die

Verknüpfung ist dabei von vornherein festgelegt. Auch hier handelt es sich wie im Fall der US 4,870,302 A um Datenverbindungen, d.h. um einen Austausch von Daten.

- 5 Da die Programmierbarkeit in Form eines Speichers mit Umsetzung in geschaltete Datenpfade recht teuer ist, wie im Fall des Rechners aber bislang das komplette Programm im Baustein lagert, erreichen die PLDs trotz Potenzials für hohe funktionale Dichte "nur" den Faktor 10 mehr als Prozessoren. Das be-
10 deutet, dass der Geschwindigkeitsgewinn in PLDs gegenüber Prozessorlösungen teuer erkaufte ist.

- Aufgabe der vorliegenden Erfindung ist es deshalb, die programmierbare Logikeinrichtung mit den eingangs genannten
15 Merkmalen dahingehend auszugestalten, dass bei hoher funktionseller Dichte eine hohe Geschwindigkeit des PLDs mit einfachen Mitteln zu erreichen ist.

- Diese Aufgabe wird erfindungsgemäß mit den in Anspruch 1 angegebenen Maßnahmen gelöst. Diese Maßnahmen umfassen eine Rekonfigurierbarkeit der Logikblöcke während des gesamten Betriebs der Logikeinrichtung dadurch, dass die Verknüpfungsmittel zusätzlich wenigstens einen konfigurierbaren Umschalt-
20 Logikblock aufweisen, mit dem eine Konfiguration wenigstens einiger der rekonfigurierbaren Logikblöcke selbst und/oder ihrer Verbindungen untereinander und/oder ihrer Verbindungen mit der Verarbeitungseinheit und/oder ihrer Verbindungen mit der Ein-/Ausgabeeinheit erfolgt.

- 30 Die mit einer solchen Architektur verbundenen Vorteile sind insbesondere darin zu sehen, dass bei begrenzter Anzahl von nunmehr rekonfigurierbaren Logikblöcken neben einer Konfiguration der Blöcke selbst auch die Verbindungen zwischen diesen Blöcken und/oder zu externen Bausteinen wie Speichereinheiten oder Mikroprozessoren wie z.B. zu der Verarbeitungseinheit und/oder zu der Ein-/Ausgabeeinheit nicht ein für alle Male fest vorgegeben sind, sondern dass diese Verbindungen
35

mit Hilfe des zusätzlich vorgesehenen Umschalt-Logikblocks während des gesamten Betriebs bedarfsmäßig erstellt, d.h. konfiguriert werden können. Eine derartige Konfiguration mittels des Umschalt-Logikblocks kann als eine Verknüpfungsoperation der erfindungsgemäß ausgestalteten Verknüpfungsmittel angesehen werden.

Die Konfiguration kann dabei jederzeit, d.h. während der gesamten, ununterbrochenen Betriebsdauer - also nicht nur während einer Start- oder Boot-Phase - vorgenommen werden. Die erfindungsgemäßen Maßnahmen bedeuten also eine Verbindung von dem Umschalt-Logikblock zu Verknüpfungsbereichen und damit zu einer entsprechenden Konfiguration derselben. Damit wird es ermöglicht, dass von einzelnen Blöcken unterschiedliche Funktionen zu unterschiedlichen Zeitpunkten auszuführen sind; d.h., deren Ausnutzung wird entsprechend erhöht. Damit verbunden ist eine entsprechende Performance-Verbesserung der gesamten Logikeinrichtung, verglichen mit einem Mikroprozessor, bzw. kein Performance-Verlust im Vergleich zu herkömmlichen PLDs.

PLDs erhalten - wie von-Neumann-Prozessoren - zwei Arten von Informationen, Code und Daten. Der Code, der grundsätzlich die Aktionswoche bestimmt und bei PLDs Konfiguration bzw. Konfigurationscode genannt wird, wird üblicherweise vor dem eigentlichen Betrieb geladen und ist dann während des Betriebs unveränderlich. Die Konfiguration bestimmt u.a. die im Baustein aktiven Verbindungen.

Die Daten können sich während des Betriebs verändern und dadurch auch den aktuellen Verlauf der Operationen beeinflussen. Während also der Code alle möglichen Wegen beinhaltet, wird die tatsächliche Nutzung - dies entspricht dem aktuell durchlaufenden Pfad - (auch) von den Daten bestimmt.

Der Umschalt-Logikblock nach der Erfindung erhält ebenfalls Code und Daten. Das Wesentliche der Erfindung besteht nun

darin, die übrigen, Nicht-Umschalt-PLDs, zu steuern, und zwar durch Wechseln bzw. Modifizieren des Codes.

Bei dem genannten Stand der Technik sowie bei den in der Praxis üblichen PLDs werden konfigurierbare Blöcke dadurch miteinander verbunden, dass die (Laufzeit-variablen) Daten austauschbar sind, also etwa in der Form, dass Datenausgänge des einen Blocks mit Dateneingängen des anderen verbunden sind. Bei dem erfindungsgemäßen Umschalt-Logikblock ist es jedoch so, dass die Ausgänge dieses Logikblocks zumindest partiell an den in anderen Ausführungsformen im Betrieb unzugänglichen Codebereich der konfigurierbaren Logikblöcke angeschlossen sind. Demgegenüber bezieht sich der Stand der Technik auf Ausführungsformen, bei denen die konfigurierbaren Logikblöcke im Datenpfad koppeln, nicht jedoch auf den Code Einfluss haben.

Vorteilhafte Ausgestaltungen der erfindungsgemäßen Logikeinrichtung gehen aus den abhängigen Ansprüchen hervor.

So kann bevorzugt der Umschalt-Logikblock in einer Ebene ausgebildet sein, die von einer Ebene mit den rekonfigurierbaren Logikblöcken verschieden ist. Unter einer verschiedenen Ebene des Umschalt-Logikblocks wird dabei jede Ebene verstanden, die nicht gleichzeitig die Ebene der rekonfigurierbaren Logikblöcke ist. D.h., die Ebene des Umschalt-Logikblocks kann über, neben oder unter der Ebene der rekonfigurierbaren Logikblöcke liegen. Dabei können diese Ebenen vorteilhaft zumindest weitgehend äquivalent aufgebaut sein.

Weiterhin ist es von Vorteil, wenn die Konfiguration wenigstens einiger der rekonfigurierbaren Logikblöcke einem vorgegebenen Kontext entsprechend erfolgt.

Ferner kann bevorzugt der Umschalt-Logikblock wenigstens einen Zustandsspeicher aufweisen, der Informationen bezüglich der Funktionen der einzelnen rekonfigurierbaren Logikblöcke

enthält, so dass die Konfiguration der ausgewählten rekonfigurierbaren Logikblöcke gemäß den Funktionsinformationen des aktuellen Zustands erfolgt.

- 5 Die den erfindungsgemäßen Maßnahmen zu Grunde liegenden Überlegungen werden nachfolgend unter Bezugnahme auf die Zeichnung noch weiter erläutert. Dabei zeigen
- deren Figur 1 die Grundstrukturen von vier einfachen, bekannten Automatentypen,
- 10 deren Figur 2 die Grundstruktur einer sogenannten "Sequential Finite State Machine",
- deren Figur 3 den schematischen Aufbau einer partiell rekonfigurierbaren PLD
- und
- 15 deren Figur 4 den schematischen Aufbau einer erfindungsgemäßen Logikeinrichtung.

Im Folgenden werden aus den Maschinen vom sogenannten "Finite State"-Typ die sogenannten "Sequential Finite State Machines"

20 (SFSM) abgeleitet. Das hierfür aufgezeigte Modell ist dazu geeignet, eine Sequenz von Konfigurationen im PLD zu definieren, und genau diese Sequenz kann ohne Performance-Verlust eine wesentlich höhere funktionale Dichte (gleichbedeutend mit drastisch gesenkten Kosten für Herstellung und Betrieb)

25 eines PLDs erzeugen.

Abschnitt I ("Sequential Finite State Machines")

Eines der "klassischen" Denkmodelle für eine Hardwareentwicklung besteht in den einfachen endlichen Automaten, im Folgenden "Finite State Machines" (FSM) genannt.

30

Dieses eng mit theoretischen Konzepten (insbesondere des sogenannten „Deterministischen endlichen Automaten“) verwandte

35 Modell ist exakt wie folgt definiert:

Definition 1:

Eine FSM besteht aus einem 6-Tupel $\{A, X, Y, f, g, a_0\}$.
 $A = \{a_0, a_1, \dots, a_M\}$ ist hierbei die endliche Menge der
 Zustände, wobei a_0 den Startzustand bedeutet.
 $X = \{X_1, \dots, X_K\}$ ist die endliche Menge der Eingangsvektoren mit $X_I = (x_1, \dots, x_L)_i$, wobei $x_l \in \{0, 1, -\}$.
 $Y = \{Y_1, \dots, Y_N\}$ ist die endliche Menge der Ausgangsvektoren mit $Y_J = (y_1, \dots, y_H)_j$, wobei $y_h \in \{0, 1, -\}$.
 $f: A \times X \rightarrow A$ heißt Transitionsfunktion (Next State Decoder),
 $g: A \times X \rightarrow Y$ heißt Ausgangsfunktion (Output Decoder),
 $t = \text{Zeiteinheit}$.

Die in dieser Definition genannten Funktionen werden durch
 Schaltnetze realisiert, die den algorithmischen Zusammenhang
 zwischen den (im Wesentlichen binären) Eingangs- und Zu-
 standsvektoren darstellen. Hierzu ist allerdings notwendig,
 dass auch die Zustände binär codiert werden, was in Definitio-
 on 1 noch nicht der Fall war.

Für die Zustandsautomaten werden weiterhin drei wesentliche
 Untertypen (siehe Figur 1) unterschieden, deren Einfluss auf
 die Komplexität der Funktionen in der Praxis gegeben ist. Ein
 Entwickler wird folgende Schritte durchführen:

1. *Festlegen der Signale*: Das Schaltwerk wird als „Black Box“ mit den erforderlichen Eingangs- und Ausgangssignalen skizziert.
2. *Entwerfen des Zustandsdiagramms*: Dieser Schritt ist der eigentliche Kern der Synthese, da hier das zu lösende Problem formal beschrieben wird. Als Zustandsdiagramm sind Zustandsgraphen, Programmablaufpläne oder auch Schaltwerkstabellen möglich.
3. *Aufstellen der Schaltwerkstabelle* als formaler Ausgangspunkt für alle weiteren Operationen.

4. *Zustandsminimierung*: Die Minimierung der Anzahl der Zustände soll eine Vereinfachung des Designs erreichen.
 5. *Zustandscodierung*: Bei synchronen Schaltwerken mit synchronisierten Eingängen (nicht für Mealy-Automat gemäß Figur 1d) lassen sich beliebige Codierungen für die Zustände aus Z angeben. Beim Medwedjew-Automat gemäß Figur 1b müssen die Codierungen allerdings mit den gewünschten Ausgangssignalen übereinstimmen; beim Moore-Automat gemäß Figur 1c hingegen können die Codierungen so gewählt werden, dass sich vereinfachte Schaltnetze für Next State Decoder (f) und Output Decoder (g) ergeben.
 6. *Berechnung von Folgezustands- und Ausgangsschaltnetzen*: Zur konkreten Berechnung des Folgezustandsschaltnetzes muss ein Register- bzw. Flipfloptyp gewählt werden, da deren Eingänge (T, D, RS oder JK) durch dieses Schaltnetz belegt werden, aber unterschiedliche Funktionalitäten zeigen.
 7. *Realisierung des Schaltwerks und Test*
- 20 Aus der Praxis weiß man nun, dass die Implementierung als einfache Maschine - man könnte sie auch als „flache Maschine“ bezeichnen - nicht unbedingt die beste ist. Ein kooperierender Automat, bestehend aus mehreren einfachen Automaten, die miteinander gekoppelt sind, kann dies häufig wesentlich besser im Sinne von flächeneffizienter. Zudem sollte bedacht werden, dass der vorstehend skizzierte Designfluss nicht unbedingt in der angegebenen Weise durchgeführt werden muss. Gerade mit zunehmender Beschreibung in Hochsprachen wie VHDL ist eine Hinwendung zu mehr algorithmischem Stil zu erkennen.
- 30 Unabhängig davon soll jedoch die FSM als das grundlegende Modell angenommen werden. Man kann sich nun vorstellen, nicht nur eine FSM zu haben, sondern mehrere, von denen exakt eine zu einem Zeitpunkt aktiv ist. Ein ausgezeichneter Teil (der sogenannte „Sequencer“) schaltet dann in Abhängigkeit von Eingangssignalen oder erreichten Resultaten zwischen den einzelnen FSMs um.
- 35

Ausgehend von dieser FSM wird nachfolgend die Sequential Finite State Machine (SFSM) korrekt definiert und dargestellt :

5 Definition 2:

Eine Sequential Finite State Machine (SFSM) besteht aus einem 5-Tupel (B, B_0, C, V, h) . Hierbei stellt $B = \{B_0, \dots B_k\}$ eine endliche Menge von Finite State Machines (FSM) dar, B_0 ist die Start-FSM. $C = \{C_0, \dots C_k\}$ beschreibt eine endliche Menge von Zuständen zur Kenntlichmachung der aktuellen FSM. $V = \{V_1, \dots V_N\}$ ist die endliche Menge der (zusätzlichen) Eingangsvektoren mit $V_i = (v_1, \dots v_L)_i$, wobei $v_1 \in \{0, 1, -\}$. $h: B \times V \rightarrow B$ heißt FSM-Transitionsfunktion (Next FSM Decoder).

Der wesentliche Vorteil dieser SFSM (vgl. hierzu auch Figur 2) liegt erst einmal in der Modellierung. Ein Designer hat die Chance, sein Design in kleinere Portionen zu teilen. In der technischen Ausführung wird man dann versuchen, die (weiterhin endliche) Menge der Zustände aus allen FSMs B_k auf eine einheitliche Zustandscodierung abzubilden. Zusätzlich hierzu muss eine Codierung für die C_k der aktuellen FSM B_k mitgeführt werden, um für die Funktion h die Berechnung der nächsten FSM zu ermöglichen.

Andererseits zeigt ein Blick auf Figur 2, dass eigentlich nur wenig gewonnen ist. Falls der Takt für alle Register identisch ist, dann wurden in Figur 1a die Register und das Schaltnetz $f(u^*, x^*)$ (Next State Decoder), * bedeutet Gesamtmenge) in Teilmengen von Registern mit $f(u, x, c)$ und $h(u, v, c)$ eingeteilt, die in der Realisierung wieder zusammengeführt werden. Minimaler wird die Lösung dadurch kaum, denn in einem PLD heutiger Bauart müssen alle Teile, also alle Teil-FSM, auf dem Baustein integriert werden. Hier setzen nun die erfindungsgemäßen Überlegungen an.

Es existieren zwar wenige FPGAs („Field-Programmable Gate Arrays“; vgl. z.B. „Spektrum der Wissenschaft“, August 1997, Seiten 44 bis 49), die dynamisch rekonfigurierbar sind, jedoch nur partiell. Diese rekonfigurierbare Eigenschaft könnte man ausnutzen, indem in einem permanenten Teil $h(u, v, c)$ und die Register zur FSM-Codierung sowie im nachladbaren Teil die aktuelle FSM geladen und ausgeführt werden. Das Problem hierbei ist der Performanceverlust beim Nachladen. Dies ist aus dem nachfolgenden Beispiel ersichtlich:

Beispiel

Die gern genommene Straßenverkehrsampel als Beispiel für eine Finite State Machine kann auch sehr gut als Beispiel für eine Sequential FSM genommen werden, wenn man an einen Nachtmodus denkt. Das Wort Modus zeigt schon den Weg: Im Umschalten zwischen einzelnen FSMs sollte so etwas wie ein Moduswechsel liegen, und Tag- und Nachtmodus schließen sich nun einmal gegenseitig aus.

FSM1 integriert nun die Tagampel, FSM2 die Nachtampel (z.B. gelbes Blinken für die Nebenstraßen), und im übergeordneten Sequencer wird an Hand eines Zeitsignals entschieden, welche FSM zur Ausführung kommt und welche ruht. Der Sequencer wird als FSM0 implementiert.

Die Beschreibung für hierfür erforderliche drei FSM (0 .. 2) wird getrennt durchgeführt, dann aber auf einem PLD integriert, so dass im Bereich 0 die FSM0, 1 FSM1 und 2 FSM2 liegen würde. Ein Flächengewinn wäre damit nicht zu verzeichnen.

Im nachladbaren Fall müsste FSM0 permanent zur Verfügung stehen, außerdem ein Bereich, der das Maximum von {FSM1, FSM2} aufnehmen könnte. In diesen Bereich wäre dann ständig eine von zweien geladen, und das Umschalten würde eine Rekonfiguration nach sich ziehen.

Zur Verdeutlichung des Vorteils bei dynamisch rekonfigurierbaren PLDs, wie sie Logikeinrichtungen nach der Erfindung darstellen, wird die Übergangsfunktion $f(u, x, c)$ (vgl. Figur 2) nunmehr als $f_c(u, x)$ (Äquivalentes gilt für Funktion g) bezeichnet. Mit dieser Darstellung soll die Unterschiedlichkeit in der Abhängigkeit demonstriert werden: u und x bewirken eine ständige, mit dem Takt verbundene Fortentwicklung des Zustands der Maschine, während c die Abhängigkeit vom wesentlich selteneren Moduswechsel beschreibt.

Die Rekonfiguration selbst würde vergleichsweise viel Zeit in Anspruch nehmen; es wurde nun erkannt, dass eine wesentlich günstigere Lösung in einer erweiterten Architektur von PLDs besteht, in denen zugleich mehrere Programme dynamisch umschaltbar gespeichert sind. Eine entsprechende Logikeinrichtung könnte als Multi-Plane-PLD bezeichnet werden; um jedoch dem Aspekt des jeweiligen „Kontext“ Rechnung zu tragen und darzustellen, dass auch nur Teilbereiche von Umschaltvorgängen betroffen sind, wird hier der Name „Dynamical Multi-Context PLDs (dMC-PLDs)“ gewählt.

Abschnitt II (Erfindungsgemäße Architektur von „Dynamical Multi-Context PLDs“)

Zunächst sei ein Problem angesprochen, dass sowohl für klassische dynamisch rekonfigurierbare als auch für dMC-PLDs gilt. Wird nämlich lediglich ein Teilbereich umgeschaltet, müssen dessen Schnittstellen nach außen hin, also insbesondere in Richtung der verbleibenden Teile, konstant bleiben. Zugleich sind einige Funktionsprogramme (sogenannte „Routingressourcen“) von dem Umschalten mit betroffen, da innerhalb des umzuschaltenden Blocks nicht nur Logik, sondern auch Verbindungen liegen müssen.

Konsequenterweise wird im Folgenden für die erfindungsgemäße Logikeinrichtung ein PLD-Block als eine rekonfigurierbare

Einheit, bestehend aus logischen Verarbeitungseinheiten mit Routingressourcen und Schnittstellen (sogenanntes „Routing“) zu anderen Blöcken definiert; diese Definition stimmt mit den in der Praxis üblichen, partiell rekonfigurierbaren Bausteinen überein. Die Definition dieses Blocks weicht dabei von der üblichen Definition der logischen Blöcke ab.

Wenn man sich jetzt einmal einen fiktiven PLD mit solchen Blöcken anschaut, erhält man eine Architektur, wie sie aus Figur 3 hervorgeht. In der Figur sind bezeichnet mit 2 eine partiell rekonfigurierbare PLD-Einrichtung bzw. -Struktur, mit 3A bis 3D vier PLD-Blöcke (= Logikblöcke), mit 4 ein (globales) Routing, das eine Verarbeitungseinheit mit Funktionsprogrammen und Schnittstellen zu den Blöcken bildet, sowie mit 5 ein I/O-Bereich, der Mittel zum Verknüpfen der Logikblöcke mit der Verarbeitungseinheit und einer Ein-/und Ausgabeinheit darstellt. Man kann sich diese Architektur so vorstellen, dass – einmal abgesehen von den I/O-Ressourcen 5 am Außenrand – jeder PLD-Block 3A bis 3D einem vollwertigen CPLD („Complex PLD“) oder FPGA entspricht. Derartige Architekturen existieren bereits, z.B. „Cypress Delta39k“, „Lattice-Vantis Godfather“-Architektur; sie sind jedoch nicht im Betriebszustand der PLD-Einrichtung rekonfigurierbar.

Bei der in Figur 4 schematisch angedeuteten dMC-PLD-Architektur einer Logikeinrichtung 7 nach der Erfindung wird die PLD-Struktur 2 nach Figur 3 durch einen zusätzlichen PLD überlagert. Dieser als Umschalt-Logikblock 8 bezeichnete Extra-PLD ist bei bekannten Logikeinrichtungen nicht vorhanden und soll die Umschaltfunktion $h(u, v, c)$ (vgl. Figur 2) übernehmen. Er stellt folglich einen Teil von Verknüpfungsmitteln dar, mit denen die Konfiguration wenigstens einiger der Logikblöcke 3A bis 3D untereinander und/oder mit der Verarbeitungseinheit 4 und/oder der Ein-/Ausgabeinheit 5 erfolgt. Dies bedeutet, dass die Ausgänge in Figur 4 einem gespeicherten Kontext c entsprechen und die Auswahl der aktuellen Funktionen $f_c(u, x)$ und $g_c(u, x)$ steuern.

Für diesen Extra-PLD ergibt sich damit eine Struktur, die aus vielen Eingängen (u, x) wenige, ausschließlich gespeicherte Ausgänge berechnet.

5

Die sich aus diesem Aufbau ergebenden Vorteile sind darin zu sehen, dass die funktionale Dichte und damit die Ausnutzbarkeit gesteigert werden bzw. die Kosten pro Anwendung zu senken sind. Folgende grobe Abschätzung kann dafür gegeben werden:

10

Es sei angenommen, dass die Ausnutzung der verwendeten Transistoren 1:20 ist; d.h., von 20 eingesetzten Transistoren ist tatsächlich nur einer in der (programmierbaren) logischen Funktion sichtbar. Für den Extra-PLD 8 lassen sich 20 % Overhead veranschlagen, da es sich um einen Logikblock handelt, der nicht zur eigentlichen Funktionalität beiträgt, sondern nur die anderen Blöcke 3A bis 3D umschaltet. Für die Speicherung und Decodierung der zusätzlichen Programme (es werden lediglich 4 gespeicherte Funktionen pro PLD-Block angenommen) werden weitere 6 Transistoren berechnet (4 Transistoren zur Speicherung und 2 zur Auswahl 1 aus 4). Daraus ergibt sich eine neue Ausnutzung von 4:30, mithin eine Steigerung der Dichte gegenüber bekannten Ausführungsformen um

15

20

25

Gegebenenfalls könnte sich eine Schwierigkeit dadurch ergeben, dass sich auf Grund mehrerer Konfigurationen, aus denen eine aktuelle auszuwählen ist, der kritische Pfad innerhalb eines PLD-Blocks um einen Transistor verlängern kann, was mit der Erniedrigung der maximalen Taktfrequenz einhergehen würde. Hierzu muss zwischen sogenannter SRAM-basierter Speicherung einerseits und sogenannter FLASH-EEPROM bzw. Anti-Fuse andererseits unterschieden werden.

30

35

Für SRAM-basierte Verfahren wird mit der US 6,011,740 A ein Verfahren vorgeschlagen, die Speicherung mehrerer Konfigura-

tionen in einem Schatten-RAM z.B. als Ringspeicher zu realisieren und die aktuell benötigte Information in eine Arbeitszelle zu laden. Dieses Verfahren garantiert eine maximale Arbeitsgeschwindigkeit.

5

Für Flash- und Antifuse-Speicherung könnte als eigentliche Zelle ebenfalls eine nachladbare SRAM-Zelle in Betracht kommen, so dass die obengenannte Lösung zum Tragen kommt; dies bedeutet jedoch einen zusätzlichen Aufwand und möglicherweise eine Verringerung der Geschwindigkeit, bedingt durch den Technologiewechsel bei der Speicherung. Bei diesen Technologien wird tatsächlich ein weiterer Path-Transistor, der den aktuellen Kontext decodiert, im Pfad eingeführt, gegebenenfalls in Form von Dual-Gate-FETs.

10
15

Die genaue Einteilung, welche Blockgröße und Extra-PLD-Größe für die Vielzahl an Anwendungen von Vorteil sind, kann durch Simulationen an Beispielen ermittelt werden.

20 Zu einer Realisierung einer erfindungsgemäßen Logikeinrichtung nach Figur 4 kann man sich vorstellen, dass die beiden schematisch dargestellten Schichten, nämlich die mit dem Extra-PLD 8 (= Umschalt-Logikblock) und die mit den Logikblöcken 3A bis 3D, jeweils in einer Ebene E2 bzw. E1 und bevorzugt
25 zumindest weitgehend äquivalent sind und sich nicht nur gegenseitig umschalten, sondern vielmehr auch die Programmierung des jeweils anderen Teils erzeugen. Umschaltung bedeutet in diesem Fall, dass zwischen vorbestimmten Programmen ausgewählt wird. Gegenseitige Programmierung hingegen kann zur Anpassung (Evolution) an die jeweilige Umgebung genutzt werden.
30 Ein einfaches Beispiel kann eine digitale PLL sein, die eine Frequenz mittels Laufzeiteffekten an einen Mastertakt anpasst.

Patentansprüche

1. Einrichtung der programmierbaren Logik
 - mit mehreren Logikblöcken mit konfigurierbaren Eigenschaften, die jeweils mindestens eine Verarbeitungseinheit mit Funktionsprogrammen und Schnittstellen zu den jeweils anderen Logikblöcken umfassen,
 - mit mindestens einer den Logikblöcken zugeordneten Ein-/Ausgabeeinheit,
 - 10 und
 - mit Mitteln zum Verknüpfen der Logikblöcke
 - a) untereinander,
 - b) mit mindestens einer der Verarbeitungseinheiten eines anderen Logikblocks
 - 15 und
 - c) mit der mindestens einen Ein-/Ausgabeeinheit,
- g e k e n n z e i c h n e t durch eine Rekonfigurierbarkeit der Logikblöcke (3A bis 3D) während des gesamten Betriebs der Logikeinrichtung (7) dadurch, dass die Verknüpfungsmittel zusätzlich wenigstens einen konfigurierbaren Umschalt-
- 20 Logikblock (8) aufweisen, mit dem eine Konfiguration wenigstens einiger der rekonfigurierbaren Logikblöcke (3A bis 3D) selbst und/oder ihrer Verbindungen untereinander und/oder ihrer Verbindungen mit der mindestens einen Verarbeitungsein-
- 25 heit (4) und/oder ihrer Verbindungen der mindestens einen Ein-/Ausgabeeinheit (5) erfolgt.
2. Einrichtung nach Anspruch 1, d a d u r c h g e -
k e n n z e i c h n e t , dass der Umschalt-Logikblock (8)
 - 30 in einer Ebene (E2) ausgebildet ist, die von einer Ebene (E1) mit den rekonfigurierbaren Logikblöcken (3A bis 3D) verschieden ist.
3. Einrichtung nach Anspruch 2, d a d u r c h g e -
k e n n z e i c h n e t , dass die Ebenen (E1, E2) zumin-
 - 35 dest weitgehend äquivalent aufgebaut sind.

4. Einrichtung nach einem der vorangehenden Ansprüche,
d a d u r c h g e k e n n z e i c h n e t , dass die Kon-
figuration wenigstens einiger der rekonfigurierbaren Logik-
blöcke (3A bis 3D) einem vorgegebenen Kontext (c) entspre-
5 chend erfolgt.

5. Einrichtung nach einem der vorangehenden Ansprüche, d a -
d u r c h g e k e n n z e i c h n e t , dass der Um-
schalt-Logikblock (8) wenigstens einen Zustandsspeicher auf-
10 weist, der Informationen bezüglich der Funktionen der einzel-
nen rekonfigurierbaren Logikblöcke (3A bis 3D) enthält, und
dass die Konfiguration der ausgewählten rekonfigurierbaren
Logikblöcke gemäß den Funktionsinformationen des ausgewählten
Zustands erfolgt.

15

1/3

FIG 1a

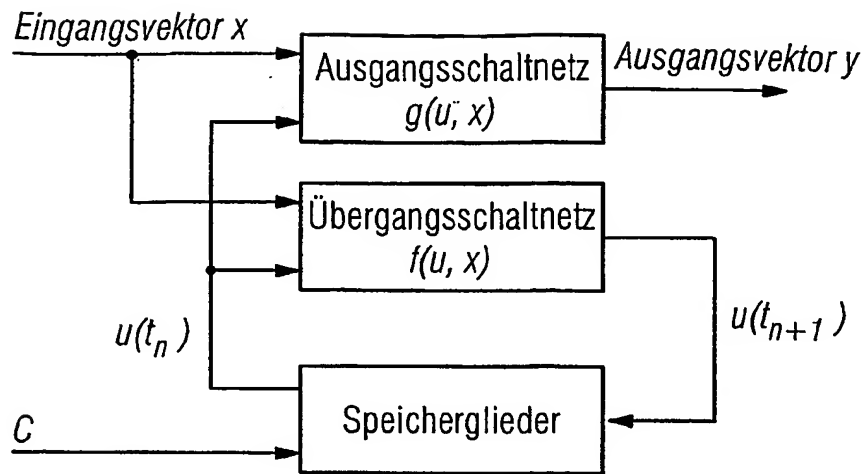


FIG 1b

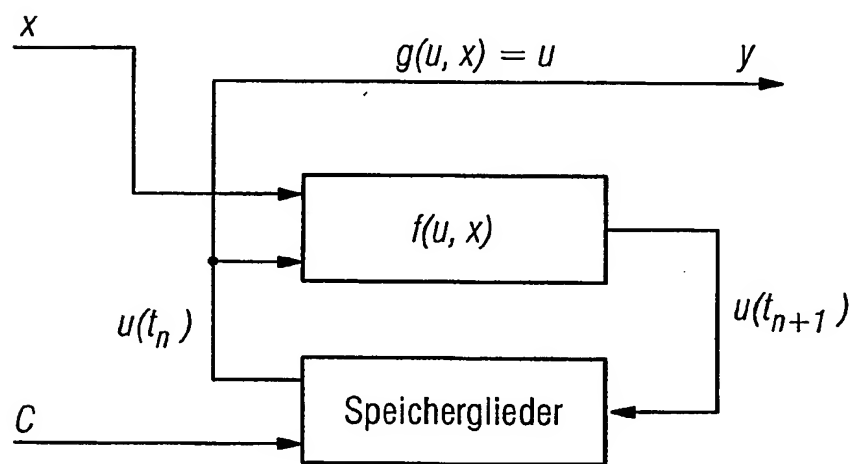
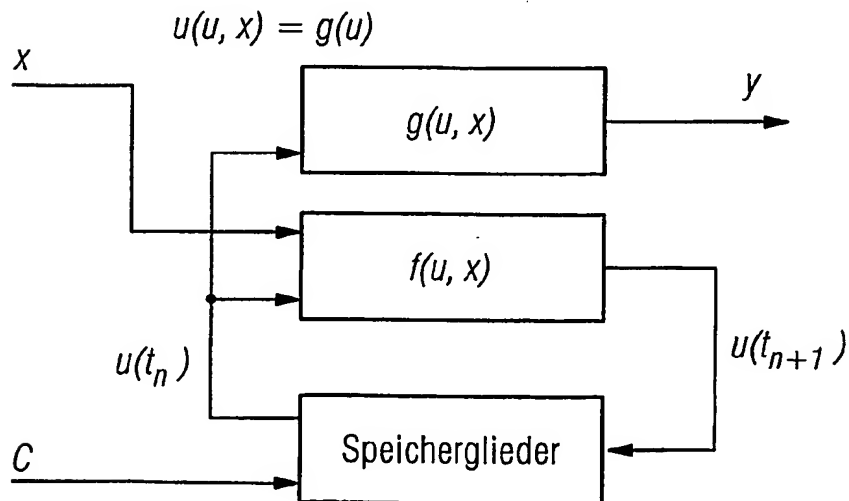


FIG 1c



2/3

FIG 1d

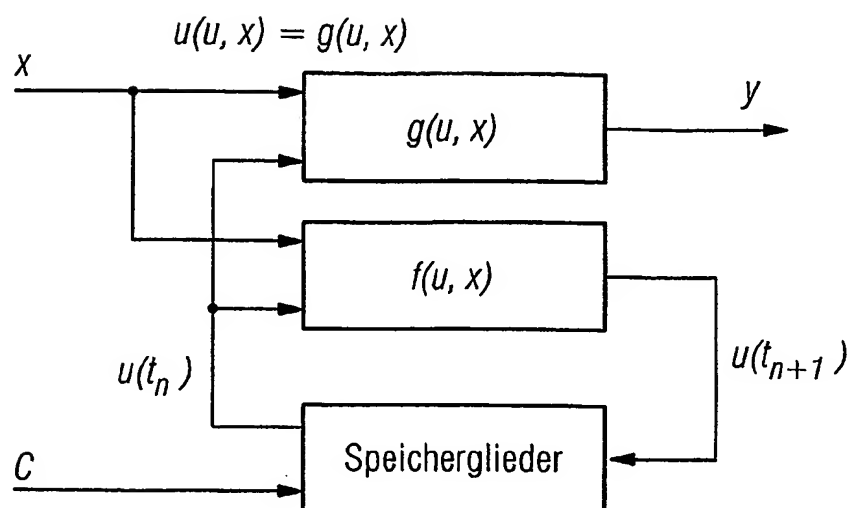


FIG 2

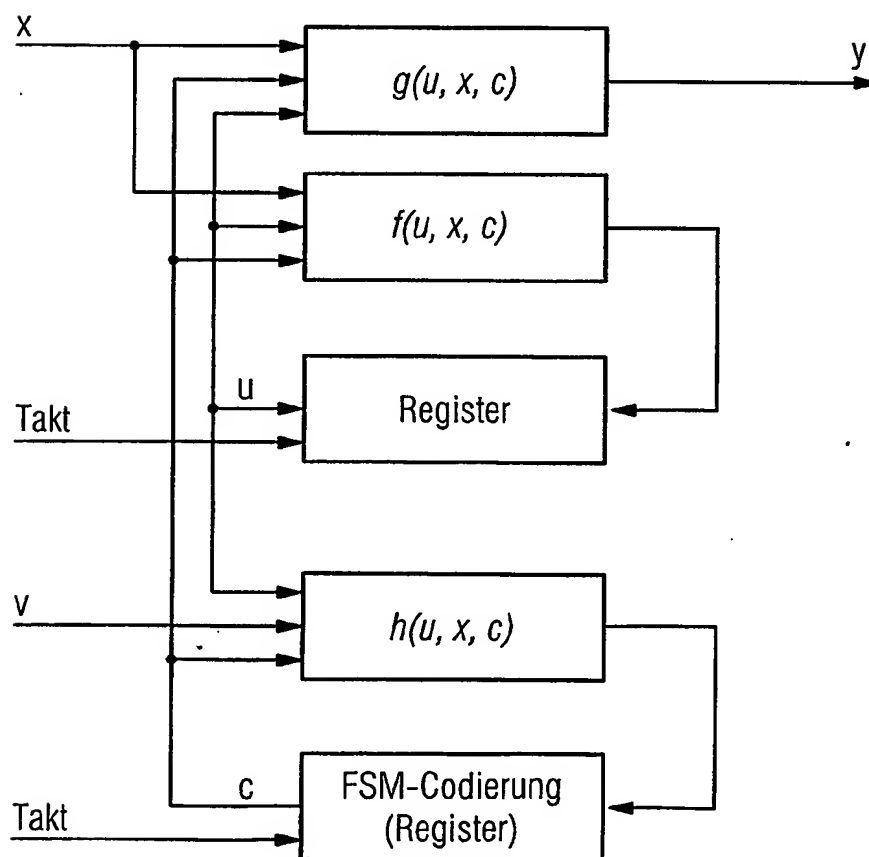


FIG 3

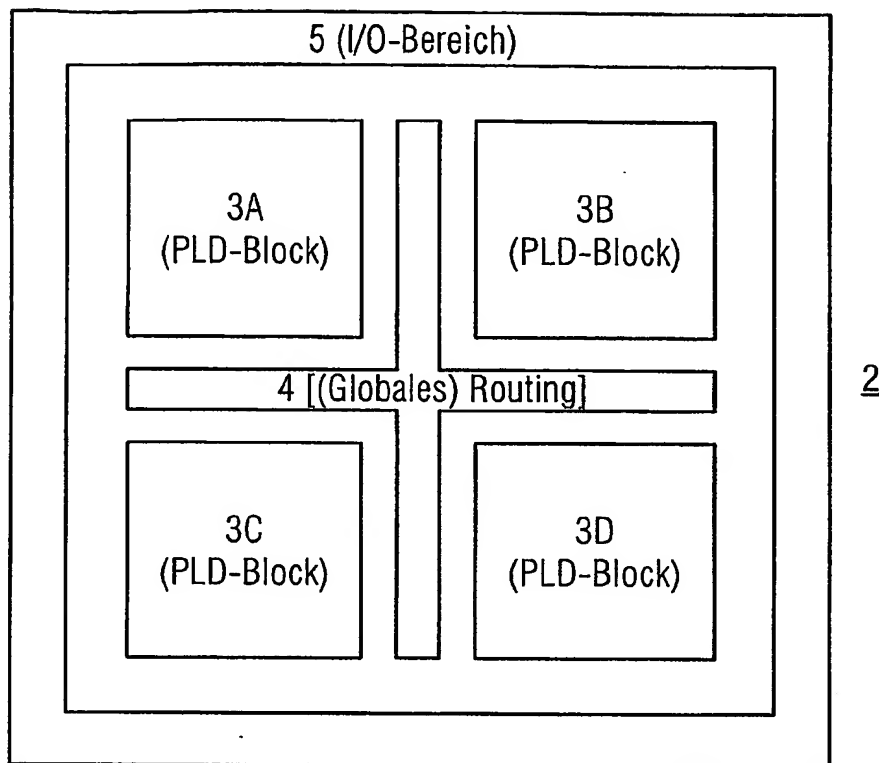
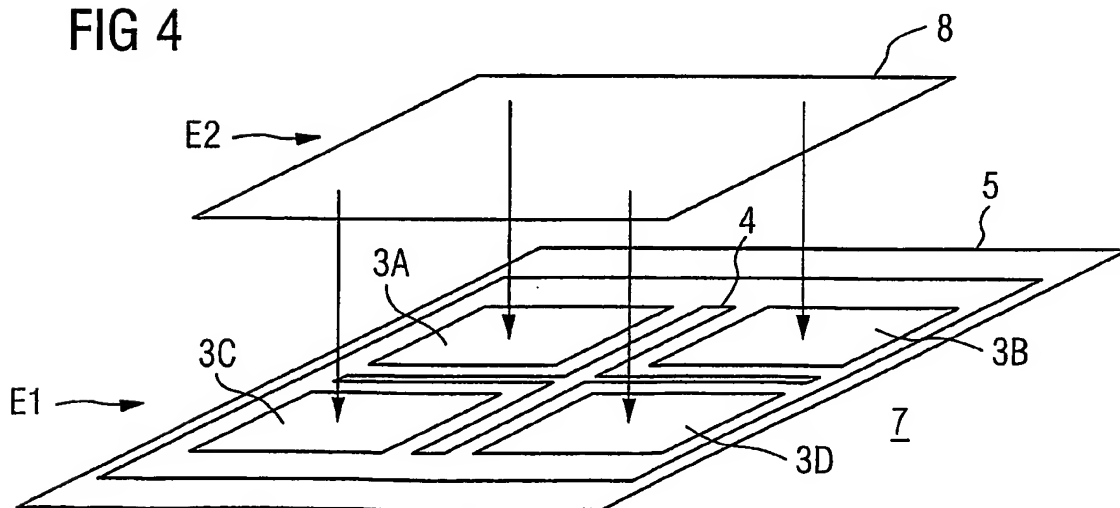


FIG 4

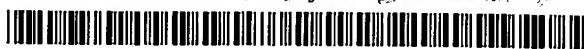


(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES
PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro



10/532642



(43) Internationales Veröffentlichungsdatum
13. Mai 2004 (13.05.2004)

PCT

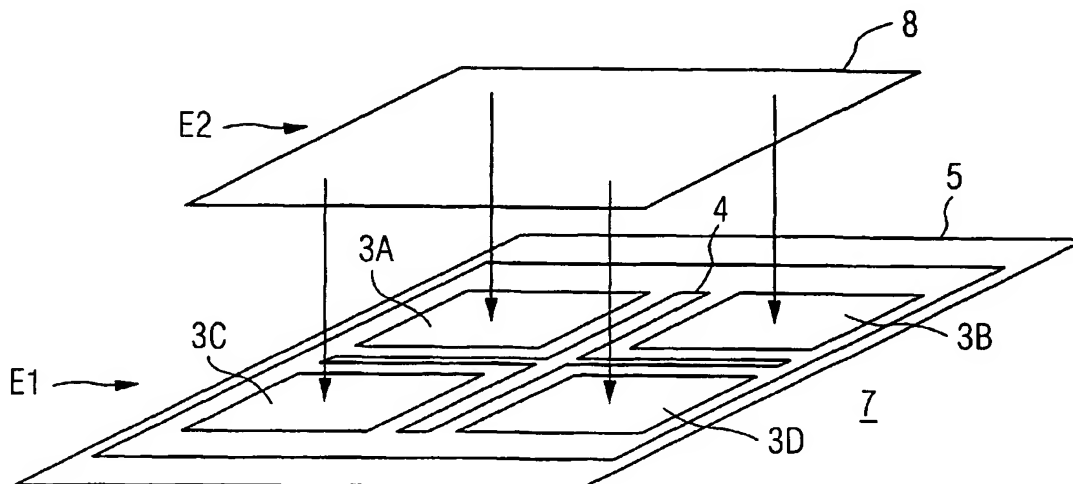
(10) Internationale Veröffentlichungsnummer
WO 2004/040766 A3

- (51) Internationale Patentklassifikation⁷: H03K 19/177
- (21) Internationales Aktenzeichen: PCT/DE2003/003524
- (22) Internationales Anmeldedatum:
23. Oktober 2003 (23.10.2003)
- (25) Einreichungssprache: Deutsch
- (26) Veröffentlichungssprache: Deutsch
- (30) Angaben zur Priorität:
102 49 676.5 24. Oktober 2002 (24.10.2002) DE
- (71) Anmelder (für alle Bestimmungsstaaten mit Ausnahme von
US): SIEMENS AKTIENGESELLSCHAFT [DE/DE];
Wittelsbacherplatz 2, 80333 München (DE).
- (72) Erfinder; und
- (75) Erfinder/Anmelder (nur für US): SIEMERS, Christian
[DE/DE]; Mary-Cassatt-Ring 38, 38446 Wolfsburg (DE).
- (74) Gemeinsamer Vertreter: SIEMENS AKTIENGE-
SELLSCHAFT; Postfach 22 16 34, 80506 München
(DE).
- (81) Bestimmungsstaaten (national): CN, JP, KR, US.
- (84) Bestimmungsstaaten (regional): europäisches Patent (AT,
BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR,
HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR).
- Veröffentlicht:
— mit internationalem Recherchenbericht

[Fortsetzung auf der nächsten Seite]

(54) Title: PROGRAMMABLE LOGIC DEVICE

(54) Bezeichnung: EINRICHTUNG DER PROGRAMMIERBAREN LOGIK



(57) Abstract: The invention relates to a programmable logic device (7) comprising several logic blocks (3A to 3D) with configurable characteristics, elements for linking the logic blocks to one another and a processing unit (4) and an input/output unit (5). In addition, the linking elements have at least one configurable changeover logic block, enabling the logic blocks (3A to 3D) to be re-configured during the operation of the logic device (7). Said changeover logic block is used for the configuration of at least one of the re-configurable logic blocks (3A to 3D) and its connection to other blocks and/or its connection to the processing unit (4) and/or to the input/output unit (5).

(57) Zusammenfassung: Die Einrichtung (7) der programmierbaren Logik umfasst mehrere Logikblöcke (3A bis 3D) mit konfigurierbaren Eigenschaften und Mittel zum Verknüpfen der Logikblöcke untereinander und mit einer Verarbeitungseinheit (4) und einer Ein-/Ausgabeeinheit (5). Eine Rekonfigurierbarkeit der Logikblöcke (3A bis 3D) während des Betriebs der Logikeinrichtung (7) ist dadurch gegeben, dass die Verknüpfungsmittel zusätzlich wenigstens einen konfigurierbaren Umschalt-Logikblock (8) aufweisen, mit dem die Konfiguration wenigstens einiger der rekonfigurierbaren Logikblöcke (3A bis 3D) untereinander und/oder mit der Verarbeitungseinheit (4) und/oder der Ein-/Ausgabeeinheit (5) erfolgt.

WO 2004/040766 A3



— vor Ablauf der für Änderungen der Ansprüche geltenden Frist; Veröffentlichung wird wiederholt, falls Änderungen eintreffen

Zur Erklärung der Zweibuchstaben-Codes und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

(88) Veröffentlichungsdatum des internationalen

Recherchenberichts:

28. Oktober 2004

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen

DE 03/03524

A. KLASSTIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
IPK 7 H03K19/177

Nach der Internationalen Patentklassifikation (IPK) oder nach der nationalen Klassifikation und der IPK

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
IPK 7 H03K

Recherchierte aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)

EPO-Internal, WPI Data

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	TESSIER R ET AL: "RECONFIGURABLE COMPUTING FOR DIGITAL SIGNAL PROCESSING: A SURVEY" JOURNAL OF VLSI SIGNAL PROCESSING SYSTEMS FOR SIGNAL, IMAGE, AND VIDEO TECHNOLOGY, KLUWER ACADEMIC PUBLISHERS, DORDRECHT, NL, Bd. 28, Nr. 1/2, Mai 2001 (2001-05), Seiten 7-27, XP001116960 ISSN: 0922-5773 Seiten 21-22; Abbildungen 6,7 -/-	1-5



Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen



Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :

A Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist

E älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist

L Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)

O Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht

P Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist

T Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist

X Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden

Y Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist

Z Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche

7. September 2004

Absendedatum des internationalen Recherchenberichts

21/09/2004

Name und Postanschrift der Internationalen Recherchenbehörde
Europäisches Patentamt, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Bevollmächtigter Bediensteter

Oloff, H

C.(Fortsetzung) ALS WESENTLICHE GEGEHENE UNTERLAGEN

Kategorie°	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	<p>SIEMERS C ET AL: "Reconfigurable computing based on universal configurable blocks-a new approach for supporting performance- and realtime-dominated applications"</p> <p>COMPUTER ARCHITECTURE CONFERENCE, 2000. ACAC 2000. 5TH AUSTRALASIAN CANBERRA, ACT, AUSTRALIA 31 JAN.-3 FEB. 2000, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 31. Januar 2000 (2000-01-31), Seiten 82-89, XP010370824 ISBN: 0-7695-0512-0 Paragraphen 4.3 und 4.4, Abbildungen 4,5</p>	1-5
X	<p>SKLYAROV V: "Reconfigurable models of finite state machines and their implementation in FPGAs"</p> <p>JOURNAL OF SYSTEMS ARCHITECTURE, ELSEVIER SCIENCE PUBLISHERS BV., AMSTERDAM, NL, Bd. 47, Nr. 14-15, August 2002 (2002-08), Seiten 1043-1064, XP004375020 ISSN: 1383-7621 Paragraphen 6 mit 10 Abbildungen 8c, 10</p>	1-5
A	<p>CHIEN A A ET AL: "MORPH: a system architecture for robust high performance using customization (an NSF 100 TeraOps point design study)"</p> <p>FRONTIERS OF MASSIVELY PARALLEL COMPUTING, 1996. PROCEEDINGS FRONTIERS '96., SIXTH SYMPOSIUM ON THE ANNAPOLIS, MA, USA 27-31 OCT. 1996, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 27. Oktober 1996 (1996-10-27), Seiten 336-345, XP010201607 ISBN: 0-8186-7551-9 das ganze Dokument</p>	1-5
A	<p>RABAEY J M: "Hybrid reconfigurable processors-the road to low-power consumption"</p> <p>VLSI DESIGN, 1998. PROCEEDINGS., 1998 ELEVENTH INTERNATIONAL CONFERENCE ON CHENNAI, INDIA 4-7 JAN. 1998, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 4. Januar 1998 (1998-01-04), Seiten 300-303, XP010263449 ISBN: 0-8186-8224-8 Abschnitt "4. THE BERKELEY PLEIADES PROJECT", Abbildung 5</p>	1-5

-/--

C.(Fortsetzung) ALS WESENTLICH GEGESEHENE UNTERLAGEN		
Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
A	DEVADAS S ET AL: "DECOMPOSITION AND FACTORIZATION OF SEQUENTIAL FINITE STATE MACHINES" IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, IEEE INC. NEW YORK, US, Bd. 8, Nr. 11, 1. November 1989 (1989-11-01), Seiten 1206-1217, XP000126892 ISSN: 0278-0070 das ganze Dokument	1-5
A	US 2002/143505 A1 (DRUSINSKY DORON) 3. Oktober 2002 (2002-10-03) in der Anmeldung erwähnt Zusammenfassung	1-5
X	WO 00/69072 A (MORPHICS TECH INC) 16. November 2000 (2000-11-16) Zusammenfassung; Anspruch 1; Abbildungen 1,2,8,9	1-5
A	US 4 870 302 A (FREEMAN ROSS H) 26. September 1989 (1989-09-26) in der Anmeldung erwähnt Zusammenfassung	1-5
P,X	P.M. HEYSTERS, J.M. SMIT, B. MOLENKAMP: "Reconfigurable Architecture for Handheld Devices" PROCEEDINGS OF THE 3D PROGRESS WORKSHOP ON EMBEDDED SYSTEMS, 24. Oktober 2002 (2002-10-24), XP002295073 UTRECHT das ganze Dokument	1-5
P,X	C. SIEMERS: "Configurable Computing - Ansätze, Chancen und Herausforderungen" TAGUNGSBAND EMBEDDED WORLD 2003, Februar 2003 (2003-02), Seiten 631-648, XP002295074 Abschnitt "4 Entwicklungsmethodik für Reconfigurable Computing", Abbildung 16	1-5
A	M KÖSTER, J. TEICH: "(Self-)reconfigurable Finite State Machines: Theory and Implementation" PROCEEDINGS OF THE 2002 DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, 4. März 2002 (2002-03-04), Seiten 1-8, XP002295075 PARIS das ganze Dokument	1-5

-/--

C.(Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Beitr. Anspruch Nr.
X	C. SIEMERS, S. WENNEKERS: "The Universal Configurable Block/Machine System - An Approach for a Homogeneous Configurable SoC-Architecture" PROCEEDINGS OF THE WORKSHOP HETEROGENEOUS RECONFIGURABLE SYSTEMS ON CHIP - SOC -, Februar 2002 (2002-02), Seiten 1-6, XP002295076 HAMBURG Abbildung 4	1-5
A	E. CANTO ET AL.: "A Temporal Bipartitioning Algorithm for Dynamically Reconfigurable FPGAs" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION - VLSI - SYSTEMS, Bd. 9, Nr. 1, Februar 2001 (2001-02), Seiten 210-218, XP002295077 das ganze Dokument	1-5
A	HARTENSTEIN R: "Coarse grain reconfigurable architectures" CONFERENCE PROCEEDINGS ARTICLE, 30. Januar 2001 (2001-01-30), Seiten 564-569, XP010537867 das ganze Dokument	1-5

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen

DE 03/03524

Im Recherchenbericht angeführtes Patentdokument	Datum der Veröffentlichung	Mitglied(er) der Patentfamilie	Datum der Veröffentlichung
US 2002143505 A1	03-10-2002	KEINE	
WO 0069072 A	16-11-2000	AU 5127500 A	21-11-2000
		CA 2371077 A1	16-11-2000
		EP 1177630 A1	06-02-2002
		JP 2002544699 T	24-12-2002
		WO 0069072 A1	16-11-2000
		US 6333641 B1	25-12-2001
US 4870302 A	26-09-1989	US RE34363 E	31-08-1993

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER: _____**

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.